WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 4, Number 2, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v4n2.18
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Research on Real-time E-commerce Price Comparison System Using Python Web Scraping Technology

Fan Chen *

[1] Philippine Christian University, Philippine Christian University 1648 Taft Ave, Malate, Manila, 1004 Mrtro Manila, Philippine

[2] School of Computer Engineering, Jingchu University of Technology, Jingmen 448000, China

ABSTRACT

This paper presents a study on the development of a real-time price comparison system for e-commerce platforms using Python-based web scraping technologies. The objective is to analyze and compare product prices across major e-commerce platforms, including Taobao, JD.com, and Amazon, by leveraging different web scraping libraries such as BeautifulSoup, Scrapy, and Selenium. The study outlines the technical challenges associated with dynamic content extraction, anti-scraping mechanisms, and the effectiveness of various scraping tools. Through experimental evaluation, we compare the scraping efficiency, success rate, and data accuracy across the three platforms. The results indicate that JD.com, with its mostly static content, offers the highest scraping efficiency when using BeautifulSoup, while Amazon, despite slower page loading, achieves a high success rate using Scrapy. Taobao, on the other hand, presents significant challenges due to its dynamic loading and strict anti-scraping measures, making Selenium the most appropriate tool, albeit with slower processing times. This research provides valuable insights into the selection of optimal web scraping techniques for real-time price comparison systems and highlights the impact of platform-specific challenges on the accuracy and efficiency of data extraction.

KEYWORDS

Python; Web Scraping; E-Commerce; Price Comparison; Real-time data; BeautifulSoup; Scrapy; Selenium; Anti-scraping measures

1. INTRODUCTION

In today's rapidly evolving e-commerce landscape, price comparison has become a critical tool for both consumers and businesses. With the increasing number of online shopping platforms and the abundance of available products, consumers are continuously seeking the best deals. To facilitate this, price comparison systems have emerged, providing users with real-time pricing data across multiple platforms, thereby enhancing decision-making processes. The success of such systems relies heavily on the ability to collect accurate, real-time price information from various e-commerce platforms efficiently.

The rise of web scraping technologies has made it possible to automate the extraction of large amounts of data from websites, including product prices. Web scraping, using languages like Python, has proven to be a powerful method for gathering structured data from e-commerce platforms such as Taobao, JD.com, and Amazon. These platforms, however, vary significantly in terms of webpage architecture, content delivery mechanisms, and anti-scraping measures, posing challenges for real-time data extraction.

Python's versatility and rich ecosystem of web scraping libraries, such as BeautifulSoup, Scrapy, and Selenium, offer diverse approaches for navigating these challenges. BeautifulSoup is well-suited for extracting data from static HTML content, Scrapy excels at high-performance, large-scale data crawling, and Selenium is effective for interacting with dynamically loaded content. However, the effectiveness of these libraries is highly dependent on the characteristics of the target platforms.

Despite its potential, web scraping for real-time price comparison presents several challenges, including dynamic content rendering, CAPTCHA verifications, and IP blocking, which are commonly employed by e-commerce platforms to prevent automated data extraction. Thus, the selection of appropriate tools and strategies is crucial to ensure the accuracy, efficiency, and sustainability of the scraping process.

This study aims to develop a real-time price comparison system by leveraging different Python web scraping libraries and to evaluate their performance across three major e-commerce platforms—Taobao, JD.com, and Amazon. By comparing the scraping efficiency, success rate, and data accuracy, this research seeks to identify the most suitable techniques for real-time price extraction and to provide practical insights into overcoming platform-specific challenges. The findings of this study are expected to contribute to the development of more efficient and robust e-commerce price comparison systems, benefiting both consumers and businesses in making more informed decisions.

2. DEVELOPMENT HISTORY

2.1. The Development and Application of Python Web Scraping Technology

Web scraping technology was initially used for collecting and analyzing information from the internet. Armbrust et al. (2010) were the first to propose a cloud computing-based web scraping framework, which addressed performance issues related to large-scale data collection. With the rise of e-commerce platforms, web scraping technology became widely used for the automated collection of product information. Barrett et al. (2015) pointed out that Python, with its simple syntax and rich library ecosystem, has become the preferred language for developing web scrapers, especially for collecting and processing e-commerce data.

The commonly used Python web scraping libraries include BeautifulSoup, Scrapy, and Selenium. According to Barrett et al., BeautifulSoup is suitable for parsing static web page data, Scrapy is more appropriate for large-scale web crawling, and Selenium is capable of handling dynamically loaded content. Iansiti and Lakhani (2014) emphasized the importance of these tools in automated data collection, particularly in scenarios where no API support is available.

2.2. The Development of E-commerce Price Comparison Systems

2.2.1. The Rise and Development of E-commerce Platforms

With the rapid development of internet technologies, e-commerce platforms have gradually become one of the mainstream models for global business transactions. In the late 1990s and early 2000s, early e-commerce platforms like Amazon and eBay emerged, providing consumers with a new way of shopping and making online purchases possible. During this period, e-commerce platforms primarily operated on C2C (consumer-to-consumer) and B2C (business-to-consumer) transaction models, where products were showcased on websites, and consumers could directly purchase them online.

Entering the 21st century, with the maturity of broadband networks, mobile internet, and payment systems, the e-commerce industry entered a phase of rapid development. Chinese platforms like Taobao and Tmall, under Alibaba, along with international platforms like Amazon, eBay, and Walmart, not only increased the convenience of online shopping but also optimized the user

experience through personalized recommendations and fast delivery services. Meanwhile, the proliferation of mobile devices further propelled the mobile e-commerce trend, with many platforms developing dedicated mobile applications, allowing consumers to shop online anytime, anywhere.

2.2.2. The Development of E-commerce Price Comparison Systems

The emergence of e-commerce price comparison systems was driven by consumers' demand for finding the lowest prices. As e-commerce platforms diversified, product prices varied across platforms, and consumers often needed to spend significant time comparing prices on multiple websites. To simplify this process, price comparison systems were developed. Early price comparison systems, such as *PriceGrabber* and *Google Shopping*, connected to major e-commerce platforms via APIs, enabling automatic price updates. These systems not only helped consumers quickly find the best prices but also promoted market transparency.

However, as the number of e-commerce platforms increased and some restricted access to their APIs, API-based price comparison systems began to face challenges such as delayed data updates and limited platform coverage. To address these challenges, web scraping-based price comparison systems started to emerge. These systems use automated programs to scrape product information from web pages and update price data in real time. Compared to the traditional API connection method, web scraping can bypass API restrictions and directly retrieve publicly available data from web pages while running simultaneously across multiple platforms.

First-generation Price Comparison Systems

The earliest price comparison systems relied primarily on e-commerce platforms' APIs for data extraction and aggregation. This method played an important role in the early development of e-commerce platforms. Platforms like PriceGrabber and Shopzilla used APIs to aggregate product information from different e-commerce sites, offering product search and price comparison features. However, the limitation of such systems was that many e-commerce platforms were unwilling to provide API access, especially companies concerned that competitors or intermediary platforms might exploit their data. Moreover, API data update frequencies were relatively low, making it difficult to achieve real-time updates.

Second-generation Price Comparison Systems

As web scraping technology matured, second-generation price comparison systems gradually replaced the early API-based systems. Web scraping could directly scrape page data from e-commerce websites, enabling more comprehensive product information retrieval. Since web scrapers do not depend on APIs, they can handle multiple websites that do not provide API access. Additionally, the advent of scraping technology made real-time price extraction possible, significantly enhancing the timeliness and coverage of price comparison systems.

For example, well-known price tracking platforms like CamelCamelCamel use web scraping technology to collect historical price data from Amazon in real time, providing users with price trend analysis and helping consumers identify the best time to purchase. European price comparison platforms like Skroutz and Pricerunner also widely use web scraping technology, further enhancing the breadth and depth of data retrieval.

Third-generation Price Comparison Systems

In recent years, with the development of big data, artificial intelligence (AI), and machine learning technologies, price comparison systems have undergone a third wave of technological innovation. Modern price comparison systems can not only scrape real-time data from e-commerce platforms but also use machine learning models to predict price trends and offer personalized price recommendations. By analyzing users' browsing and purchase histories, these systems can help consumers find products and prices that better meet their needs. Additionally, combined with big data

technologies, these systems can analyze vast amounts of user purchase behavior, enabling e-commerce platforms to optimize their pricing strategies.

For example, tools like Honey automatically recognize the products users are viewing and recommend corresponding price histories or coupons. These systems not only extend the capabilities of traditional price comparison systems but also integrate real-time price monitoring, discount push notifications, and purchase decision recommendations.

2.2.3. The Current Status and Challenges of E-commerce Price Comparison Systems

Although web scraping-based price comparison systems have significantly improved the speed and accuracy of price retrieval, they also face several challenges. First, many e-commerce platforms restrict scraping access due to concerns about data protection. For example, platforms use anti-scraping techniques such as dynamic content loading, CAPTCHA verifications, and request frequency limitations to prevent unauthorized data extraction.

Second, web scraping itself has limitations, such as consuming considerable network resources, particularly when processing large amounts of dynamic web content. Although tools like Selenium can handle dynamic data loading, their efficiency is generally lower than that of scrapers designed for static web pages.

Lastly, the legality of web scraping in price comparison systems has become an increasing concern. The widespread use of scraping technology may violate the terms of service or privacy policies of some platforms, and in some countries, it may even face legal challenges. Therefore, how to maintain data extraction efficiency while adhering to relevant laws and regulations has become a key consideration for modern price comparison system developers.

3. OVERVIEW AND CONFIGURATION OF PYTHON WEB SCRAPING LIBRARIES

Due to its simple syntax and extensive support for third-party libraries, Python has become an ideal choice for developing web scrapers. Depending on various requirements and application scenarios, Python offers multiple libraries and frameworks suited for web scraping, including tools such as BeautifulSoup, Scrapy, and Selenium. Choosing the appropriate scraping library can not only enhance data extraction efficiency but also effectively handle different website structures and anti-scraping mechanisms. The following sections provide a detailed introduction to the features, usage scenarios, and configuration methods of these mainstream web scraping libraries.

3.1. BeautifulSoup

BeautifulSoup is a lightweight HTML and XML parsing library primarily used for quick parsing and data extraction of static content in web pages. Its core function is to transform complex web page structures into traversable tree structures and extract the necessary elements from them. BeautifulSoup excels at handling structured static pages, making it ideal for small-scale data scraping and page parsing tasks.

Applicable scenarios: BeautifulSoup is very suitable for handling static web pages with simple structures and no complex dynamic loading. For example, product detail pages or review pages in e-commerce platforms.

Configuration method:

(1) Install BeautifulSoup and its dependent HTML parser lxml or html. arser:

(2) Configure basic crawler environment

```
import requests
from bs4 import BeautifulSoup

url = 'https://example.com/product'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')
product_name = soup.find('h1', {'class': 'product-name'}).text
print(product_name)
```

Figure 1. Configure basic crawler environment

3.2. Scrapy

Scrapy is a powerful and efficient Python web crawling framework designed to handle large-scale web crawling tasks. The advantage of Scrapy lies in its built-in high-performance asynchronous processing mechanism, which enables it to crawl a large number of web pages in a short period of time. It also provides automated crawling, data pipeline, data storage, and scheduling functions, making it the preferred choice for building complex data crawling projects.

Applicable scenarios: Scrapy is very suitable for projects that require crawling a large amount of data on multiple websites, especially for multi page data crawling on e-commerce platforms, such as product list pages and their pagination processing.

Configuration method:

(1) Install Scrapy

(2) Create Scrapy project

(3) Write scraping scripts in the spiders directory:

```
import scrapy

class ExampleSpider(scrapy.Spider):
    name = "example_spider"
    start_urls = ['https://example.com/products']

    def parse(self, response):
        for product in response.css('div.product'):
            yield {
                'name': product.css('h2.name::text').get(),
                'price': product.css('span.price::text').get(),
            }
```

Figure 2. scraping scripts in the spiders directory

(4) Run the crawler and save the results

3.3. Selenium

Selenium is an automated testing tool commonly used for handling content crawling of dynamic web pages. It controls the browser to achieve interactive operations on web pages and can handle JavaScript rendered page elements. Therefore, in the field of web crawling, Selenium is often used to crawl web pages that require dynamic loading to obtain data.

Applicable scenarios: Selenium is suitable for processing dynamic web pages that require simulation of user behavior, such as e-commerce pages that require clicks, scrolling, or login operations to load all data.

Configuration method:

(1) Install Selenium and Chrome WebDriver:

(2) Download and configure Chrome WebDriver:

Download the corresponding version of ChromeDriver from the official website and add its path to the system variable or project.

(3) Write Selenium crawler scripts:

```
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome(executable_path='/path/to/chromedriver')
driver.get('https://example.com/product')
price = driver.find_element(By.CLASS_NAME, 'price').text
print(f"The product price is {price}")
driver.quit()
```

Figure 3. Selenium crawler scripts

4. REAL TIME CRAWLING OF PRICE INFORMATION FROM MULTIPLE E-COMMERCE PLATFORMS THROUGH WEB CRAWLING TECHNOLOGY AND COMPARISON

Capturing real-time price information on e-commerce platforms is the core task of building a price comparison system. By using Python web crawling technology, we can crawl the prices of specified products from different e-commerce platforms such as Taobao, JD.com, and Amazon, and conduct comparative analysis. This process usually includes the following steps: requesting a webpage, parsing content, extracting price information, processing data, and making comparisons.

4.1. Implementation of Real-Time Capture of Price Information

Using different crawling technologies to capture product price information for the three major platforms of Taobao, JD.com, and Amazon:

Taobao: The page structure of Taobao is complex and uses dynamic loading technology, so Selenium needs to be used to simulate user behavior and capture price data rendered through JavaScript.

JD: Most of JD's product pages are static web pages, and using BeautifulSoup can effectively extract product price information.

Amazon: The Amazon page structure is clear, and Scrapy can quickly crawl its price data, supporting efficient large-scale data scraping.

The following is the pseudocode for implementing a crawler to crawl product prices:

```
# Example: Using BeautifulSoup for JD.com to scrape product prices
import requests
from bs4 import BeautifulSoup

url_jd = 'https://item.jd.com/100012043978.html'  # JD.com product URL
response = requests.get(url_jd)
soup = BeautifulSoup(response.text, 'html.parser')
price_jd = soup.find('span', {'class': 'price'}).text
print(f"JD.com Price: {price_jd}")

# Example: Using Selenium for Taobao to scrape dynamically loaded product prices
from selenium import webdriver
from selenium.webdriver.common.by import By

driver = webdriver.Chrome(executable_path='/path/to/chromedriver')
driver.get('https://item.taobao.com/item.htm?id=100012043978')  # Taobao product URL
price_tb = driver.find_element(By.CLASS_NAME, 'tb-rmb-num').text
print(f"Taobao Price: {price_tb}")
driver.quit()

# Example: Using Scrapy for Amazon to scrape product prices
import scrapy

class AmazonSpider(scrapy.Spider):
    name = 'amazon_spider'
    start_urls = ['https://www.amazon.cn/dp/B08C1W5N87']  # Amazon product URL

    def parse(self, response):
        price_amazon = response.css('span.a-price-whole::text').get()
        print(f"Amazon Price: {price_amazon}")
```

Figure 4. the pseudocode for implementing a crawler to crawl product prices

4.2. Comparison of Product Information Retrieval Efficiency among Different E-commerce Platforms

In order to compare the efficiency of product information retrieval on different e-commerce platforms, this article conducted data retrieval experiments on the three major platforms of Taobao, JD.com, and Amazon, mainly comparing the following aspects:

(1) Crawling time: The total time required from sending a request to obtaining product price information.

(2) Capture success rate: The ratio of successfully capturing correct price information.

(3) Data accuracy: The degree to which the prices crawled by the crawler match the prices displayed on the webpage.

(4) Anti crawler measures: The platform imposes restrictions on crawlers, including CAPTCHA, request frequency limitations, etc.

In the experiment, we crawled the same product from three major platforms and evaluated efficiency through 100 crawling experiments. The experimental results are shown in the following table:

Table 1. Experimental Results

Platform	Library Used	Avg. Scraping Time (s)	Success Rate (%)	Data Accuracy (%)	Anti-scraping Measures
Taobao	Selenium	5.8	85	98	CAPTCHA, dynamic loading, IP blocking
JD.com	BeautifulSoup	1.3	95	99	Request rate limiting
Amazon	Scrapy	1.0	92	97	Request rate limiting, slow loading

4.3. Data Analysis and Comparison Results

According to the data in the table, it can be seen that:

JD has the highest efficiency in crawling product prices, with the shortest average crawling time of only 1.3 seconds. This is because most of JD's pages are static content, which can be processed quickly using BeautifulSoup, and its anti crawling mechanism is relatively loose.

Amazon comes in second place. Scrapy's asynchronous processing mechanism makes Amazon's crawling speed faster, but sometimes Amazon loads slow data, so the success rate of crawling is slightly lower.

The crawling efficiency of Taobao is relatively low, and using Selenium to simulate user operations takes a long time. In addition, the Taobao platform has strict anti crawling mechanisms, including dynamic content loading, CAPTCHA verification, and IP blocking, which significantly reduces the success rate of crawling.

By comparing different platforms, it can be seen that the choice of crawler library and platform directly affects the efficiency and success rate of data crawling. BeautifulSoup is the best choice for quickly capturing data on static pages; Selenium is more effective for dynamic pages, but its speed is slower. Scrapy is suitable for large-scale data scraping tasks, especially performing well when dealing with multiple pages. In practical applications, suitable crawler libraries should be selected based on the characteristics of specific platforms to ensure real-time crawling and data accuracy.

5. CONCLUSION

This study successfully developed an efficient, real-time e-commerce price comparison system using Python web scraping technology. The research demonstrates that Python's scraping libraries can significantly enhance the data acquisition efficiency of price comparison systems, especially in cases where e-commerce platforms do not offer API access, presenting unique advantages in data collection. The study highlights that different Python libraries have specialized strengths: BeautifulSoup is highly effective for scraping data from small-scale, static web pages, Scrapy excels in large-scale web crawling tasks due to its built-in efficiency features, and Selenium provides a robust solution for handling dynamic web content by simulating user interactions within browsers.

The findings underscore the importance of selecting the appropriate tool based on the specific requirements of each task. Moreover, the system developed in this study enables users to monitor real-time price updates across multiple platforms, which is crucial in the fast-paced, competitive e-commerce environment. This approach has practical implications for both consumers and businesses looking to optimize their pricing strategies or shopping decisions.

However, while the system achieves real-time updates and efficient data collection, certain limitations remain. For instance, the handling of dynamically loaded web content using Selenium can be resource-intensive, especially when dealing with large-scale websites or high traffic. Thus, further research should focus on optimizing performance, particularly in scenarios involving extensive

dynamic content and high data volume. Moreover, incorporating advanced technologies such as machine learning could offer predictive capabilities, allowing the system to forecast price trends, and provide personalized product recommendations based on user behavior and historical price data.

Additionally, ethical and legal considerations surrounding web scraping must be addressed in future developments. As web scraping may conflict with the terms of service of certain platforms, developers should ensure compliance with relevant data protection and privacy laws, especially as regulations around data use become stricter globally. Moving forward, balancing system performance, data integrity, and legal compliance will be crucial in ensuring that such systems are both effective and responsible.

In conclusion, the integration of Python web scraping technologies into e-commerce price comparison systems offers significant improvements in real-time data acquisition and usability. Future enhancements should focus on system scalability, performance in handling dynamic web content, and the application of predictive models, while remaining vigilant about legal and ethical data considerations.

ACKNOWLEDGEMENTS

In the process of writing and researching this paper, I received a lot of help and support from teachers, classmates, friends, and family. Here, I would like to express my sincere gratitude to them.

Firstly, I would like to express my special thanks to my supervisor for providing highly constructive suggestions on the topic selection and research methodology design of my thesis. At the critical point of the paper, he pointed out the direction of my research and helped me make breakthrough progress in the application of web crawling technology and data analysis on e-commerce platforms.

Secondly, I would like to thank all the classmates in the laboratory who discussed the implementation plan of Python crawler technology with me in the early stage of research, and put forward many valuable ideas and opinions. In addition, they assisted me in solving technical difficulties during data capture and processing, especially in the use of Selenium and dynamic web page processing, which provided me with great help.

I would like to express my gratitude to the e-commerce platform developers and relevant technical personnel who participated in this research, for providing technical support and open resources, which enabled the practical part of this paper to proceed smoothly. Their help greatly improved my efficiency in developing e-commerce data scraping and real-time price comparison systems.

Finally, I would like to express my sincere gratitude to all those who have provided me with help and support. It is your support that has enabled me to successfully complete this research. I sincerely hope that the results of this paper can not only provide some useful insights for the academic community, but also make some contributions to the development of the industry in practice.

REFERENCES

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50-58. https://doi.org/10.1145/1721654.1721672

[2] Barrett, M., Davidson, E., Prabhu, J., & Vargo, S. L. (2015). Service innovation in the digital age: Key contributions and future directions. *MIS Quarterly*, *39*(1), 135-154. https://doi.org/10.25300/MISQ/2015/39.1.07

[3] Berman, S. J., & Marshall, A. (2014). The next digital transformation: From an individual-centered to an everyone-to-everyone economy. *Strategy & Leadership*, *42*(5), 9-17. https://doi.org/10.1108/SL-07-2014-0048

[4] Chesbrough, H. (2010). Business model innovation: Opportunities and barriers. *Long Range Planning*, *43*(2-3), 354-363. https://doi.org/10.1016/j.lrp.2009.07.010

[5] Iansiti, M., & Lakhani, K. R. (2014). Digital ubiquity: How connections, sensors, and data are revolutionizing business. *Harvard Business Review*, *92*(11), 90-99.

[6] Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. *National Institute of Standards and Technology*, *53*(6), 50. https://doi.org/10.6028/NIST.SP.800-145

[7] Teece, D. J. (2010). Business models, business strategy and innovation. *Long Range Planning*, *43*(2-3), 172-194. https://doi.org/10.1016/j.lrp.2009.07.003

[8] Vargo, S. L., & Lusch, R. F. (2004). Evolving to a new dominant logic for marketing. *Journal of Marketing*, *68*(1), 1-17. https://doi.org/10.1509/jmkg.68.1.1.24036

[9] Weinberg, B. D., Parise, S., & Guinan, P. J. (2011). Social media usage and innovation: An empirical study of the role of social media in enhancing performance. *Journal of Organizational Computing and Electronic Commerce*, *21*(4), 361-382. https://doi.org/10.1080/10919392.2011.614787

[10] Zhang, X., & Li, H. (2017). Technology-Driven Innovation in E-Commerce: How Web Crawlers and Data Mining are Revolutionizing E-Commerce Platforms. *International Journal of Electronic Commerce*, *22*(3), 220-245. https://doi.org/10.1080/10864415.2017.1325706